# Software Requirements Specification

# for

# <Project>

Version 1.0 approved

Prepared by <author>

<organization>

<date created>

Table of Contents

Table of Contents [ii](#__RefHeading___Toc441230970)

Revision History [ii](#__RefHeading___Toc441230971)

1. Introduction [1](#__RefHeading___Toc441230972)

1.1 Purpose [1](#__RefHeading___Toc441230973)

1.2 Document Conventions [1](#__RefHeading___Toc441230974)

1.3 Intended Audience and Reading Suggestions [1](#__RefHeading___Toc441230975)

1.4 Product Scope [1](#__RefHeading___Toc441230976)

1.5 References [1](#__RefHeading___Toc441230977)

2. Overall Description [2](#__RefHeading___Toc441230978)

2.1 Product Perspective [2](#__RefHeading___Toc441230979)

2.2 Product Functions [2](#__RefHeading___Toc441230980)

2.3 User Classes and Characteristics [2](#__RefHeading___Toc441230981)

2.4 Operating Environment [2](#__RefHeading___Toc441230982)

2.5 Design and Implementation Constraints [2](#__RefHeading___Toc441230983)

2.6 User Documentation [2](#__RefHeading___Toc441230984)

2.7 Assumptions and Dependencies [3](#__RefHeading___Toc441230985)

3. External Interface Requirements [3](#__RefHeading___Toc441230986)

3.1 User Interfaces [3](#__RefHeading___Toc441230987)

3.2 Hardware Interfaces [3](#__RefHeading___Toc441230988)

3.3 Software Interfaces [3](#__RefHeading___Toc441230989)

3.4 Communications Interfaces [3](#__RefHeading___Toc441230990)

4. System Features [4](#__RefHeading___Toc441230991)

4.1 System Feature 1 [4](#__RefHeading___Toc441230992)

4.2 System Feature 2 (and so on) [4](#__RefHeading___Toc441230993)

5. Other Nonfunctional Requirements [4](#__RefHeading___Toc441230994)

5.1 Performance Requirements [4](#__RefHeading___Toc441230995)

5.2 Safety Requirements [5](#__RefHeading___Toc441230996)

5.3 Security Requirements [5](#__RefHeading___Toc441230997)

5.4 Software Quality Attributes [5](#__RefHeading___Toc441230998)

5.5 Business Rules [5](#__RefHeading___Toc441230999)

6. Other Requirements [5](#__RefHeading___Toc441231000)

Appendix A: Glossary [5](#__RefHeading___Toc441231001)

Appendix B: Analysis Models [5](#__RefHeading___Toc441231002)

Appendix C: To Be Determined List [6](#__RefHeading___Toc441231003)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

The purpose of this software product is to provide a method of determining the winner(s) of elections in an efficient and transparent fashion. The software is intended to

## Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

## Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

## Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

## References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# Overall Description

## Product Perspective

Diagram

Description automatically generated<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

This election voting system was developed with the intention of providing a software method of determining the results of chosen elections in a logical, efficient, and transparent fashion. It fulfills the needs of all the players involved: election officials get an unconvoluted, digital method of counting election results, the media get auto-generated statistics and metrics to report on, and audit officials get a transparent and organized view of how the election software was run.

## Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

* Run program – Loads up software and begins prompting user for ballot data and type
* Select Ballot File – User identifies ballot file to be used
* Read ballot file – System imports ballot file from election needed to be counted
* Label Ballots – System goes through ballot file and gives each ballot an ID to use in counting methods
* Indicate election type – User selects between running Instant Runoff or Open Party Listing methods on the election data
  + Determine winner in IR – System counts votes using IR method
  + Group candidates by party, including independents – System groups candidates by party in order to properly allocate votes
    - Determine winner in OPL – System counts votes using OPL method
* Display winner/results to screen – System displays the winners of the election to the screen
* Show redistribution votes of IR in audit file – System recreates flow of logic that it used to reallocate votes in the IR method
* Produce Audit file – System creates audit file containing all counted ballot IDs, and how they were distributed to candidates at all stages of the election
* Resolve Tie – In the event of a tie, the system performs a random coin flip to determine the winning candidate / party
* Produce report for media personnel – System generates report containing election metrics like vote totals, candidate / party performance and other things to report on
* Retrieve audit file – Audit official queries the system to export the audit file to a viewable format

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

* Election officials, who oversee running elections and overseeing the method of counting the votes properly. They are authorized to interact with the ballot data, and run the election software, and are thus the most crucial user class for this product
* Audit officials oversee ensuring that the election was carried out correctly and will thus use the system to get a log of how ballots were counted and distributed. These officials are authorized to interact with the system, but only to generate the audit log, not to count votes.
* Media officials interact with the system indirectly, in that they are sent reports containing the statistics about the election results. These reports are auto-generated and sent from the system to the media, so the media do not have authorization to utilize the system

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## 

## SF\_004: Indicating Election Type

4.4.1 Description: Indicating which election type the given ballot file belongs to, either instant runoff or open party listing.

4.4.2 Actor(s): Election official

4.4.3 Organizational Benefits: Allows the election official to have final control over the way that the voting system runs. On the software side, allowing for the selection between the two allows one program to fit the two election systems.

4.4.4 Frequency of Use: This feature is utilized every time a ballot requires tabulating, i.e. for every use of the election system software.

4.4.5 Trigger: The election official starts the election system software and selects the ballot file using the command line interface. The election official then is prompted to choose the election type.

4.4.6 Precondition: ~~The election official’s login information is correct and said official has authorization to run the election software. The ballot file in question exists.~~ The ballot file is loaded correctly.

4.4.7 Postcondition: A response confirming the election type is displayed on the screen. The election type in the ballot file matches the election official’s response.

4.4.8 Main Course:

1. User is shown a prompt on the CLI giving two options for the election type, I (Instant Runoff) and O (Open Party)

2. User enters one of the characters to dictate their election type

3. User presses enter

4. System confirms the choice, and displays the choice

5. System prompts the user to confirm that it should go ahead with tallying the votes with a Y/N option

5. User presses Y and then enter, and election system software commences

4.4.9 Alternate Course:

?

4.4.10 Exceptions:

EX1: Authorization fails

1. System prompts user for their User ID a second time. Authorization is processed.

2. If authorization granted, user can continue Main Course at # .

3. If authorization is not granted, provide error message

EX2: Response to Election Type prompt is not accepted (not I/O)

1. System prompts user a second time
2. If input is of correct format, continue to MC4
3. If input is again incorrect, display error message

## SF\_005: Determine winner in case of clear majority in Instant Runoff

4.5.1 Description: Determine the winner of an Instant Runoff election where a candidate earns over 50% of the first choice votes, those being made up of each ballot’s first choice candidate, without eliminating and re-allocating less popular candidates.

4.5.2 Actor(s): System

4.5.3 Organizational Benefits: A winner of the election is determined.

4.5.4 Frequency of Use: This feature is used every time a ballot is loaded with the intent of counting the votes via the instant runoff election type. Thus, every time an election official selects IR as the election type.

4.5.5 Trigger: The system records that the user confirms instant runoff as the chosen election type and commences with the counting of the ballots.

4.5.6 Precondition: Instant Runoff is the selected election type. A valid ballot file has been uploaded.

4.5.7 Postcondition: A candidate with >50% of the initial vote is determined to be the winner by the system. If there is no initial winner, system continues the instant runoff methodology of selecting a winner.

4.5.8 Main Course

1. Ballot file is opened

2. An empty data structure is created to hold all candidates and the IDs of each ballot that had said candidate as first choice, as well as said ballot’s 2nd, 3rd, 4th, and 5th choice candidates. The 2nd – 5th choice elements will be used for re-allocation of ballots whose first choice was eliminated due to being the least popular in a particular voting round

2. Ballot results are read sequentially

a. If the ballot result is for a candidate not yet in the data structure, create a new entry

b. If the ballot result is already in data structure, increment the vote count

3. Finish ballot counting

4. If there is a candidate with >50% of the total votes, determine them the winner

4.5.9 Alternate Course

?

4.5.10 Exceptions

EX1 No candidate has > 50% of the vote

1. System moves on to the elimination/re-allocation of less popular candidate vote totals as enumerated in SF\_006.

## SF\_006: Determine winner in case of no clear majority in Instant Runoff

4.6.1 Description: Sequentially eliminate less popular candidates and redistribute votes to decide a winner in the event where no candidate gets more than 50% of the votes when Instant Runoff is chosen as the election type.

4.6.2 Actor(s): System

4.6.3 Organizational Benefits: Feature accounts for possibility that no candidate achieves a popular majority of the votes. A winner of the election is determined. Follows established algorithm so that auditors can ensure a legitimate procedure was followed.

4.6.4 Frequency of Use: This feature is used every time a user runs the election software with an instant runoff ballot file and no winner is apparent after the initial votes are allocated.

4.6.5 Trigger: The system records that no candidate has earned >50% of the initial votes, and thus no popular majority has been achieved.

4.6.6 Precondition: All the ballots in the ballot file have been counted and have all been allocated to their candidate of choice.

4.6.7 Postcondition: A winner of the instant runoff election is determined.

4.6.8 Main Course

1. Candidate data structure is searched to find the candidate with the fewest votes

2. Go through the ballots attached to said candidate as their n choice, and add

them to the ballot list for the candidate listed as the n+1 choice for each ballot

3. The winner is determined to be whoever received over 50% of the vote after the re-allocation of the least popular candidate’s ballots

4.6.9 Alternate Course

?

4.6.10 Exceptions

EX1 No winner is determined after 3.

1. The 3 steps are simply run through again, starting at 1, re-allocating the next least popular candidate’s votes.

## SF\_007: Group Open Party Listing Independents into one party

4.7.1 Description: Independent candidates without an affiliated party are grouped together automatically so that the open party listing method of deciding election results can work.

4.7.2 Actor(s): System

4.7.3 Organizational Benefits: This feature allows votes for independents to be properly allocated, since they don’t have an existing party affiliation that is used to categorize votes. It allows the system to follow a generalized structure of vote allocation without having edge cases relating to interacting with independents.

4.7.4 Frequency of Use: This feature is utilized every time a ballot file is to be tabulated using the open party listing methodology. The election in question may not have any independents to group, but the system will still check every time.

4.7.5 Trigger: The user confirms that open party listing voting is to be used on the ballot data.

4.7.6 Precondition: The ballot file exists.

4.7.7 Postcondition: Independent candidates are now grouped in a data structure, allowing the open party listing voting algorithm commence.

4.7.8 Main Course

1. System creates data structure where each element contains the candidates running under the same Party affiliation

2. System goes through ballot file, and looks at candidate and party choice

3. If the party affiliation does not yet have a spot in the data structure, add one

4. If the candidate is not yet in their respective party’s slot, add them

5. If the candidate is independent, add them to a slot reserved for independents.

4.7.9 Alternate Course

4.7.10 Exceptions

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>